In the claims:

For the Examiner's convenience, all pending claims are presented below with changes shown in accordance with the new mandatory amendment format.

1-20 Cancelled

21. (Previously Presented) A method comprising:

generating a first test program population to test the functionality of an integrated circuit (IC), the first test program population comprising a plurality of test programs, each test program having a first set of instructions and data;

executing each of the test programs in the first test program population;

evaluating a first set of coverage data from the first test program population to determine if the IC has been sufficiently tested, wherein evaluating the first set of coverage data comprises comparing the coverage data to a predetermined coverage requirement; and

generating a second program population if the IC has not been sufficiently tested by the first test program population, the second test program population comprising a plurality of updated test programs, wherein each updated test program is a mutation of a test of the first test program-population for a combination of two or more of the test programs of the first test program population.

22. (Previously Presented) The method of claim 21, further comprising:

executing the second test program population.

23. (Previously Presented) The method of claim 22, wherein generating the first test program population comprises:

generating a first abstract syntax tree (AST);

generating the first set of instructions and data for the first AST; and translating the first AST into a first executable test program.

24. (Previously Presented) The method of claim 23, wherein generating the second test program population comprises:

generating a second abstract syntax tree (AST);

generating a second set of instructions and data for the second AST; and

translating the second AST into a second executable test program population.

25. (Previously Presented) The method of claim 24, further comprising mutating a selected AST.

26. (Previously Presented) The method of claim 25, wherein mutating a selected AST comprises:

selecting an AST;

removing a segment of the selected AST; and

inserting a replacement segment into the selected AST to form a mutated AST.

27. (Previously Presented) The method of claim 26, further comprising:

generating a third set of instructions and data for the mutated AST; and

translating the mutated AST into a third executable test program population.

28. (Previously Presented) The method of claim 25, wherein mutating a selected AST comprises:

selecting the first AST and the second AST; and

combining a segment of the first AST with a segment of the second AST to form a mutated AST.

29. (Previously Presented) The method of claim 28, further comprising:

generating a third set of instructions and data for the mutated AST; and

translating the mutated AST into a third executable test program population.

30. (Previously Presented) The method of claim 23, further comprising:

adding the first AST and the first set of coverage data into a test program after the first test program population has been executed.

31. (Previously Presented) A computer system comprising:

a storage device coupled to a processor and having stored therein at least one routine, which when executed by the processor, causes the processor to generate data, the routine causing the processor to,

generate a first test program population to test the functionality of an integrated circuit (IC), the first test program population comprising a plurality of test programs, each test program having a first set of instructions and data;

execute each of the test programs in the first test program population;

evaluate a first set of coverage data from the first test program population to determine if the IC has been sufficiently tested, wherein evaluating the first set of coverage data comprises comparing the coverage data to a predetermined coverage; and

generate a second program population if the IC has not been sufficiently tested by the first test program population, the second test program population comprising a plurality of updated test programs, wherein each updated test program is a mutation of a test of the first test program-population for a combination of two or more of the test programs of the first test program population.

32. (Previously Presented) The computer system of claim 31, wherein the routine further causes the processor to,

execute the second test program population.

33. (Previously Presented) The computer system of claim 32, wherein generating the first test program population comprises:

generating a first abstract syntax tree (AST);

generating the first set of instructions and data for the first AST; and

translating the first AST into a first executable test program.

34. (Previously Presented) The computer system of claim 33, wherein generating the second test program population comprises:

generating a second abstract syntax tree (AST);

generating a second set of instructions and data for the second AST; and

translating the second AST into a second executable test program population.

35. (Previously Presented) The computer system of claim 34, wherein the routine further causes the processor to mutate a selected AST.

36. (Previously Presented) The computer system of claim 35, wherein mutating a selected AST comprises:

selecting an AST;

removing a segment of the selected AST; and

inserting a replacement segment into the selected AST to form a mutated AST.

37. (Previously Presented) The computer system of claim 36, wherein the routine further causes the processor to,

generate a third set of instructions and data for the mutated AST; and

translate the mutated AST into a third executable test program population.

38. (Previously Presented) The computer system of claim 35, wherein mutating a selected AST comprises:

selecting the first AST and the second AST; and

combining a segment of the first AST with a segment of the second AST to form a mutated AST.

39. (Previously Presented) The computer system of claim 38, wherein the routine further causes the processor to,

generating a third set of instructions and data for the mutated AST; and

translating the mutated AST into a third executable test program population.

40. (Previously Presented) The computer system of claim 33, wherein the routine further causes the processor to,

add the first AST the first set of coverage data into test program population after the first test program has been executed.

41. (Previously Presented) A validation test system comprising:

a test builder to generate test program populations to test the functionality of an integrated circuit (IC);

a test generator to translate the test program populations into an executable test;

a test analyzer to execute the test program populations; and

a feedback engine to build and update a population of test programs by generating an abstract syntax tree (AST) for each test program populations.

42. (Previously Presented) The system of claim 41, wherein the feedback engine determines whether a predetermined test program population threshold has been reached after a test program populations has been executed.

43. (Previously Presented) The system of claim 42, wherein the feedback engine generates one or more mutated ASTs if it is determined that the predetermined test program population threshold has been reached.

44. (Previously Presented) The system of claim 43, wherein the feedback engine generates a mutated AST by selecting a first AST, removing a segment of the first AST and inserting a replacement segment into the first AST.

45. (Previously Presented) The system of claim 43, wherein the feedback engine generates a mutated AST by selecting a first AST and a second AST and combining a segment of the first AST with a segment of the second AST to form.